Theoretical Computer Science 115 (1993) 277–290
Elsevier

Hard promise problems and nonuniform complexity

Luc Longpré

College of Computer Science, Northeastern University, Boston, MA 02115, USA

Alan L. Selman*

Department of Computer Science, State University of New York at Buffalo, 226 Bell Hall, Buffalo, NY 14260, USA

Communicated by P. Young
Received April 1991
Revised November 1991

Abstract

Longpré, L. and A.L. Selman, Hard promise problems and nonuniform complexity, Theoretical Computer Science 115 (1993) 277–290.

For every recursive set A, let PP-A denote the following promise problem:

input x and y
promise $(x\in A)\oplus(y\in A)$
property $x\in A$.

We show that if L is a solution of PP-A, then $A\in \mathrm{P}^{L}/\mathrm{Poly}$. From this result, it follows that if A is $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-hard for NP, then all solutions of PP-A are hard for NP under a reduction that generalizes both $\leqslant_{\mathrm{T}}^{\mathrm{P}}$ and $\leqslant_{\mathrm{T}}^{\mathrm{SN}}$. Specifically, if A is NP-hard, then all solutions of PP-A are *generalized high*$_2$ (Balcázar et al., 1986). The main theorem that leads to this result states that if A is a self-reducible set and $A\in \mathrm{P}^{L}/\mathrm{Poly}$, then $\Sigma_2^{\mathrm{P},A}\subseteq\Sigma_2^{\mathrm{P},L}$. Several interesting connections between uniform and nonuniform complexity follow directly from this theorem.

1. Introduction

A promise problem is a formulation of a partial-decision problem. Informally, a promise problem has the structure

Correspondence to: L. Longpré, College of Computer Science, Northeastern University, Boston, MA 02115, USA.

*Funding for this research was provided by the National Security Agency under grant MDA-87-H-2020 and by the National Science Foundation under grant CCR-9002292.

input x

promise $Q(x)$

property $R(x)$,

where Q and R are predicates. An algorithm solves the promise problem if, given an input x, it answers the question whether $R(x)$ *given* that $Q(x)$. The behavior of such an algorithm may be arbitrary on instances x for which the promise Q is false. Promise problems are important to theoretical groundings of public-key cryptography [4, 7] and arise in combinatorial studies as well [6, 20].

In [19], the second author initiated a project to develop a theory of hard promise problems, somewhat akin to the theory that was developed over the years for NP-complete decision problems. A general framework was given. The methodology introduced was the same as the familiar one for decision problems – to show that a promise problem is hard, reduce a known hard promise problem to it. To this end, a particular class of promise problems was invented: for every set A, PP-A is the promise problem

input x and y

promise $(x\in A)\oplus(y\in A)$

property $x\in A$.

An algorithm solves this promise problem if, given input strings x and y, it answers the question whether $x\in A$ given that exactly one of x and y is in A.

The principal theorem proved in [19] is that if A is $\leqslant_d^{\mathrm{P}}$-equivalent to a disjunctive-self-reducible set in NP, then it is as hard to solve PP-A as it is to recognize A. In particular, for every $\leqslant_d^{\mathrm{P}}$-complete set A, every solution of PP-A is NP-hard. Thus, these promise problems form a class of known hard promise problems. In order to prove that some promise problem (Q, R) is NP-hard, it suffices to reduce one of these promise problems to (Q, R).

The techniques of [19] apply only when $A\in$NP and, at that, only when A is $\leqslant_d^{\mathrm{P}}$-equivalent to a disjunctive-self-reducible set. In this paper we address the following questions: What can be said of the solutions of PP-A when A is $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-complete for NP? And what can be said when A is $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-hard for NP (i.e. NP-hard)? Our results will show that solutions are "hard" in both cases. Specifically, we show that if A is NP-hard, then all solutions of PP-A are *generalized high*$_2$ (i.e. $\Sigma_3^{\mathrm{P}}\subseteq\Sigma_2^{\mathrm{P},A}$. Definitions will be given in the next section.)

This paper is about more than hard promise problems. It is also about connections between uniform and nonuniform complexity. The main theorem that leads to the result just cited is Theorem 3.10, which states that if A is a self-reducible set and $A\in P^L/\mathrm{Poly}$, then $\Sigma_2^{\mathrm{P},A}\subseteq\Sigma_2^{\mathrm{P},L}$. Let B be any self-reducible complete set. Observe that if A is taken to be any NP-hard set, and $A\in\mathrm{P}^L/\mathrm{Poly}$, then $B\leqslant_{\mathrm{T}}^{\mathrm{P}}A$; so, $B\in\mathrm{P}^L/\mathrm{Poly}$. Thus, $\Sigma_2^{\mathrm{P},B}=\Sigma_3^{\mathrm{P}}\subseteq\Sigma_2^{\mathrm{P},L}$ and, so, L is *generalized high*$_2$. As a corollary, if A is NP-hard,

$A \in \mathrm{P}^L/\mathrm{Poly}$, and L is *generalized low*$_2$, then the polynomial hierarchy must collapse to Σ_2^{p}. This corollary (Corollary 3.15) strengthens a result of [1]. The general paradigm illustrated here is to use Theorem 3.10 to show that certain sets are *generalized high*. It follows directly that such sets cannot be *generalized low* unless the polynomial hierarchy collapses.

2. Preliminaries

2.1. Promise problems

Promise problems were first described in [5]; for an in-depth treatment, the reader is referred to [7].

A promise problem is a formulation of a partial-decision problem. Informally, a promise problem has the structure

input x

promise $Q(x)$

property $R(x)$,

where Q and R are predicates. Formally, a recursive promise problem is a pair of recursive predicates (Q, R). A deterministic Turing machine M that halts on every input *solves* (Q, R) if

$$\forall x\,[Q(x) \rightarrow [M(x) = \text{“yes”} \leftrightarrow R(x)]].$$

If M solves (Q, R), then the language $L(M)$ accepted by M is a *solution* to (Q, R). So, a solution is any recursive set which agrees with R on the promise Q (regardless of what R “says” on the complement of Q). Thus, the solutions of (Q, R) are *exactly* the recursive sets of the form $(Q \cap R) \cup X$, where $X \subseteq \bar{Q}$. In particular, $Q \cap R$, R, and $\overline{Q - R} = (Q \cap R) \cup \bar{Q}$ are all solutions to (Q, R).

A set A is defined in [17] to be *p-selective* if there is a function $f: \Sigma^* \times \Sigma^* \mapsto \Sigma^*$ that satisfies each of the following:

(1) f is computable in polynomial time,

(2) $f(x, y) = x$ or $f(x, y) = x$,

(3) $x \in A$ or $y \in A \rightarrow f(x, y) \in A$.

The function f is called a *selector* for A.

The following proposition formalizes an observation in [19].

Proposition 2.1. *A set A is p-selective if and only if the promise problem* PP-A *has a solution in* P.

Proof. If A is a p-selective set with selector f, then $\{(x, y) \mid f(x, y) = x\}$ is a solution in P of PP-A. Conversely, if L is a solution in P of PP-A, then the function f defined by

$$f(x, y) = \begin{cases} x & \text{if } (x, y) \in L, \\ y & \text{otherwise} \end{cases}$$

is a selector for A. □

2.2. Self-reducibility

We will be interested in promise problems PP-A when some self-reducible set is reducible to A. Thus, we recall the definition of self-reducibility due to Meyer and Paterson [13].

Definition 2.2. A polynomial-time-recognizable partial order $<$ on Σ^* is *OK* if and only if

(i) every strictly decreasing chain is finite, and there is a polynomial p such that every finite $<$-decreasing chain is shorter than p of the length of its maximum element, and

(ii) $x < y$ implies $|x| \leqslant q(|y|)$, for some polynomial q, and all x and y in Σ^*.

Definition 2.3. A set A is *self-reducible* if and only if there is an *OK* partial order and a query machine M such that M accepts A in polynomial time with oracle A and, moreover, on any input x in Σ^*, M asks its oracle only about words strictly less than x in the partial order.

The definition of Meyer and Paterson [13] does not require an OK ordering to be polynomial-time-recognizable, but this proviso suffices for all known examples and applications in the literature. We require this condition in the proof of Theorem 3.10.

A number of specialized polynomial-time-bounded reducibilities have been studied [11]; of these we will refer to *disjunctive* reducibility, $\leqslant_{\mathrm{d}}^{\mathrm{P}}$. A simple characterization follows: $A \leqslant_{\mathrm{d}}^{\mathrm{P}} B$ if and only if there is a polynomial-time-computable function

$$f: \{0, 1\}^* \mapsto (\# \{0, 1\}^*)^*$$

such that, for each input word x, $x \in A$ if and only if $\{y_1, \ldots, y_k\} \cap B \neq \emptyset$, where $f(x) = \# y_1 \cdots \# y_k$.

A set is *d-self-reducible* if it is self-reducible and the query machine M also provides a $\leqslant_{\mathrm{d}}^{\mathrm{P}}$-reduction. This means that, on every input word x, the query machine M either

(i) decides membership of x in A in polynomial time without queries to the oracle, or

(ii) computes a set of queries $\{z_1, \ldots, z_k\}$ in polynomial time so that

$$x \in A \Leftrightarrow \{z_1, \ldots, z_k\} \cap A \neq 0.$$

2.3. Generalized lowness and generalized highness

In [3], Schöning's *low* and *high* hierarchies [14] were lifted out of NP by defining *generalized low* and *generalized high* hierarchies. A set A is defined to be *generalized low*$_n$ if $\Sigma_n^{\mathrm{P},A} \subseteq \Sigma_n^{\mathrm{P}}$, and A is defined to be *generalized high*$_n$ if $\Sigma_{n+1}^{\mathrm{P}} \subseteq \Sigma_n^{\mathrm{P},A}$. It is useful for us to understand these hierarchies in terms of reducibilities. (The following observations derive from polynomial analogs of reducibilities defined in [16] and are taken in the present form nearly directly from a paper by Schöning [15].) For each $n \geqslant 0$, define the reducibility R_n^{P} by

$$A R_n^{\mathrm{P}} B \text{ if and only if } \Sigma_n^{\mathrm{P},A} \subseteq \Sigma_n^{\mathrm{P},B}.$$

R_0^{P} is the same as polynomial-time Turing reducibility, $\leqslant_{\mathrm{T}}^{\mathrm{P}}$. R_1^{P} is the same as polynomial-time strong nondeterministic reducibility, $\leqslant_{\mathrm{T}}^{\mathrm{SN}}$ [12]. The relations R_n^{P} are reflexive and transitive and, so, $R_n^{\mathrm{P}} \cap (R_n^{\mathrm{P}})^{-1}$ is an equivalence relation.

Observe that a set A is R_n^{P}-equivalent to $\emptyset$ if and only if $\Sigma_n^{\mathrm{P},A} \subseteq \Sigma_n^{\mathrm{P}}$. Thus, a set belongs to the zero degree of the R_n^{P} reducibility if and only if it is *generalized low*$_n$. In the next section, when we prove, for some n, that a set A is *generalized low*$_n$, we will do so by demonstrating that, for some $L \in P$, $A R_n^{\mathrm{P}} L$. Also observe that a set A is R_n^{P}-hard for NP if and only if $\Sigma_n^{\mathrm{P},\mathrm{SAT}} = \Sigma_{n+1}^{\mathrm{P}} \subseteq \Sigma_n^{\mathrm{P},A}$. Thus, a set is R_n^{P}-hard for NP if and only if it is *generalized high*$_n$. It is straightforward to observe that a set cannot be both *generalized low*$_n$ and *generalized high*$_n$ unless the polynomial hierarchy collapses to Σ_n^{P}. Also, if a set is *generalized low*$_n$ (*generalized high*$_n$), for some n, then it is *generalized low*$_{n+1}$ (*generalized high*$_{n+1}$, respectively).

It follows from the observations of the previous paragraph that *generalized high* is a generalization to R_n^{P} of NP-hardness. Similarly, *generalized low* is an easiness notion.

A set is *high*$_n$ in the sense of Schöning [14] if and only if it is *generalized high* and belongs to NP. Thus, *high*$_n$ is a generalization to R_n^{P} of the NP-completeness notion. Similarly, a set is *low*$_n$ if it is *generalized low*$_n$ and belongs to NP.

It is easy to see that every *generalized low*$_n$ set belongs to Σ_n^{P}. Thus, *generalized low*$_1$ and *low*$_1$ are identical. Namely, a set A is *generalized low*$_1$ if $\mathrm{NP}^A = \mathrm{NP}$, that is, if and only if $A \in \mathrm{NP} \cap \text{co-NP}$.

Part of this work will involve an excursion into Karp–Lipton type results [8]. Here we note only the following: Let A and L be arbitrary sets. Standard techniques show that $A \in \mathrm{P}^L/\mathrm{Poly}$ if and only if there is a sparse set S such that $A \leqslant_{\mathrm{T}}^{\mathrm{P}} L \oplus S$ if and only if A has a family of polynomial-size circuits, where the circuits contain nodes that are oracle calls to L.

The power of the *generalized high* and *generalized low* concepts lies in that they enable Karp–Lipton type results to be obtained without the technical difficulties apparent in the original proofs [1, 8]. This has been noted before by Ko and Schöning [10].

3. Results

Theorem 3.1. *If L is a solution of* PP-A, *then there is a sparse set $S \subseteq A$ such that $A \leqslant_T^P L \oplus \text{prefix}(S)$.*

Proof. Let L be a solution of PP-A. Let $L_1 = \{(x, y) \mid (x, y) \in L \wedge (y, x) \notin L\}$. Note that L_1 is the set L where we remove the pairs (x, y) such that both (x, y) and (y, x) are in L. Because the changes can occur only for pairs where the promise is false, L_1 is also a solution of PP-A.

We will construct a sparse set S such that

$$x \in A \leftrightarrow (\exists z \in S)[|z| = |x| \text{ and } (z, x) \notin L_1] \tag{1}$$

Since $L_1 \leqslant_T^P L$ and S can be reconstructed from its prefixes, we can conclude $A \leqslant_T^P L \oplus \text{prefix}(S)$.

The set S will be $\bigcup S_n$, where $S_n \subseteq \Sigma^n$. Each finite set S_n is constructed in stages, each stage putting one element into S_n.

For each integer n, build a directed graph whose nodes are the strings of size n and whose edges are defined by L_1, i.e. $(x, y) \in G \leftrightarrow (x, y) \in L_1$. Form the graph G_1 by restricting G to the elements of A. If there is a node y_1 in G_1 that has no successor in G_1, then we have the relation $(y_1, x) \in L_1 \leftrightarrow x \in \bar{A}$; so, this element y_1 can be used to decide membership in A. But such a node y_1 may not exist. So, we choose the node y_1 that has the fewest successors in G_1. Now the problem is to modify the algorithm to work correctly on the successors of y_1 in G_1. To do this, we can repeat this procedure on just those members of A which are successors of y_1.

More formally, for stage i, choose the node y_i in G_i which has the fewest number of successors. Put y_i into S. Remove y_i and all the nodes that are not successors of y_i from G_i, together with all the edges incident to these nodes, to form G_{i+1}. Continue building S_n by stages until G_i is empty.

If the graph G_i has k nodes, it has at most $k(k-1)/2$ edges. Then, there must be a node with fewer than $k/2$ successors; so, the number of nodes in G_{i+1} is at most half the number of nodes in G_i. Thus, there are at most n stages and S_n contains at most n strings of length n. Hence, S is sparse.

Now we can show that S satisfies equation (1). If $x \in A$, then there is an i such that $x \in G_i$ but $x \notin G_{i+1}$. Because x gets removed, we have that $(y_i, x) \notin L_1$. (Note that $(y_i, x) \notin L_1$ even if $y_i = x$.)

If $x \notin A$, then, for any string $z \in S$, we have $z \in A$; so, $(x \in A) \oplus (z \in A)$ is true. Since L_1 is a solution, we must have $(z, x) \in L_1$. □

Corollary 3.2. *If L is a solution of* PP-A, *then* $A \in \text{P}^L/\text{Poly}$.

Recalling Proposition 2.1, we see that Theorem 3.1 extends Ko's result [9] that all p-selective sets are in P/Poly. The proof of Theorem 3.1 is actually less complicated than the proof in [9].

Corollary 3.3. (Ko [9]). *If A is a p-selective set, then $A \in \mathrm{P/Poly}$.*

Theorem 3.4. *If A is in Σ_i^{P} and B is a solution of* PP-A, *then $A\, R_{i+1}^{\mathrm{P}}\, B$* (i.e. $\Sigma_{i+1}^{\mathrm{P},A} \subseteq \Sigma_{i+1}^{\mathrm{P},B}$).

Proof. If $i=0$, the theorem is trivial. Let $i \geqslant 1$. Let C be a set in $\Sigma_{i+1}^{\mathrm{P},A}$. Then, there is a relation $R \in \mathrm{P}^A$ such that

$$x \in C \leftrightarrow \exists y_1 \forall y_2 \ldots Q y_{i+1} R(x, y_1, y_2, \ldots, y_{i+1}).$$

We have to find a new relation that uses B as oracle instead of A.

Let M be a polynomial-time oracle Turing machine such that

$$L^A(M) = \{\langle x, y_1, \ldots, y_{i+1}\rangle \mid R(x, y_1, \ldots, y_{i+1})\}.$$

Let p be a polynomial such that M uses less than $p(|x|)$ time. Let S be the sparse set given by the previous theorem. Let M' be a polynomial-time oracle Turing machine that accepts A using $B \oplus \mathrm{prefix}(S)$ as oracle. We know M' exists by the previous theorem. Let q be a polynomial such that M' uses less than $q(n)$ time.

For any finite set A, we let $c(A)$ denote an encoding of A. It is assumed that, for any string x and finite set A, deciding if $x \in A$ from $c(A)$ and x can be done in time polynomial in $|c(A)|+|x|$. For any set A and natural number n, let $A^{\leqslant n}$ denote the set of all strings in A of length less than or equal to n. Then $c(A^{\leqslant n})$ denotes the encoding of an initial segment of A.

We can create M'' such that if $l(x) = q(p(|x|))$, then

$$\langle x, y_1, \ldots, y_{i+1}, c(S^{\leqslant l(x)})\rangle \in L^B(M'') \leftrightarrow R(x, y_1, \ldots, y_{i+1}).$$

M'' simply simulates M, and whenever M makes a query, M'' simulates M'. Queries of M' are of size $\leqslant q(p(|x|))$ and can be answered by either asking directly to B or by using $c(S^{\leqslant l(x)})$.

To show that $C \in \Sigma_{i+1}^{\mathrm{P},B}$, the idea is to guess $c(S^{\leqslant l(x)})$, verify that it is correct and verify that M''^B accepts. This is formally expressed by the following predicate:

$$x \in C \leftrightarrow \exists z [(z \text{ is correct}) \text{ and } \exists y_1 \forall y_2 \ldots y_{i+1} [M''^B \text{ accepts } \langle x, y_1, \ldots, y_{i+1}, z\rangle]].$$

The second part of the "and" is a $\Sigma_{i+1}^{\mathrm{P},L}$ predicate. It remains to show that "z is correct" is a $\Sigma_{i+1}^{\mathrm{P},L}$ predicate. The predicate "$z = c(S^{\leqslant l(x)})$" is the most obvious test of correctness, but it might not be a $\Sigma_{i+1}^{\mathrm{P},L}$ predicate. The following test of correctness will be sufficient for our purpose:

$$z = c(S') \quad \text{for some } S' \subseteq A^{\leqslant l(x)}$$

and

$$\forall v \in A\, (|v| \leqslant l(x) \rightarrow (\exists w \in S')[(w, v) \notin B_1]),$$

where B_1 is as in the previous theorem: $(w, v) \in B_1 \leftrightarrow [(w, v) \in B \text{ and } (v, w) \notin B]$.

This test is a $\Sigma_{i+1}^{P,B}$ predicate, because $S' \subseteq A$ is a Σ_i^P predicate, and the second implication can be replaced by

$$\forall v: |v| \leqslant l(x) \left[v \notin A \text{ or } \left(\bigvee_{w \in S'} (w, v) \notin B_1 \right) \right],$$

which is a $\Pi_1^{P,B}$ predicate.

If the string $z = c(S^{\leqslant l(x)})$, then surely z is correct according to this definition of correctness. It remains to show that any correct string $z = c(S')$ is a valid witness.

Suppose $S' \subseteq A$. Then, because B_1 is a solution of A, we know that if $x \notin A$, then, for any $z \in A$, $(z, x) \in B_1$; so, queries to oracle A of strings not in A will always be answered correctly using $B \oplus \text{prefix}(S)$. Now, if $x \in A$, the second part of the implication in the test guarantees that the query to the oracle will also be answered correctly. This means that if S is correct according to the test above, then our simulation of queries to A using $B \oplus \text{prefix}(S)$ will be correct. □

Corollary 3.5. (Ko and Schöning [10]). *If A is in* NP *and A is p-selective, then A is low_2.*

Proof. Let $A \in \text{NP}$ be p-selective. Using Proposition 2.1, let $B \in \text{P}$ be a solution of PP-A. By Theorem 3.4, $A R_2^P B$. Since $A \in \text{NP}$, it follows from the definition that A is low_2. □

Corollary 3.6. *If $A \in \Sigma_i^P$ and A is generalized $high_{i+1}$, then every solution of* PP-A *is generalized $high_{i+1}$.*

Proof. Let C be a complete problem for NP. A *generalized $high_{i+1}$* set is a set that is R_{i+1}^P-hard for NP. Since A is *generalized $high_{i+1}$*, $C R_{i+1}^P A$. If L is a solution of PP-A, then, by the theorem, $A R_{i+1}^P L$. The corollary follows by the transitivity of the R_{i+1}^P reduction. □

Of course, the corollary applies when A is a $\leqslant_T^P$-complete set in NP. Thus, we have a specific hardness result for solutions of PP-A when A is $\leqslant_T^P$-Turing-complete for NP.

Corollary 3.7. *If A is a $\leqslant_T^P$-complete for* NP, *then every solution of* PP-A *is generalized $high_2$.*

Note that in view of Corollary 3.2 the following theorem has a weaker hypothesis and weaker conclusion than does Theorem 3.4.

Theorem 3.8. *If A is in Σ_i^P and $A \in \text{P}^L/\text{Poly}$,* then $A R_{i+2}^P L$.

The proof of this theorem is like the proof of Theorem 3.4, except that verifying the correctness of advice ("$c(S^{\leqslant l(x)})$ is correct") is now a straightforward Π_{i+1}^{P} test using only the hypothesis that A is in Σ_i^{P}. The proof is not given, for it is similar to the proof of Theorem 3.4, as well as a straightforward generalization of the following known corollary.

Corollary 3.9 (Ko and Schöning [10]). *If A is in* NP *and* $A \in \mathrm{P}/\mathrm{Poly}$, *then A is low_3.*

By Corollary 3.7, if A is $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-complete for NP, then every solution of PP-A is *generalized high*$_2$. What can be said about solutions L of PP-A when A is $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-hard for NP, but one does not assume that A belongs to NP? It turns out that the solutions are still *generalized high*$_2$, as a corollary (Corollary 3.14) of the following theorem.

Theorem 3.10. *If A is self-reducible and $A \in \mathrm{P}^L/\mathrm{Poly}$, then $A R_2^{\mathrm{P}} L$.*

Proof. Let $C \in \Sigma_2^{\mathrm{P},A}$. Then there is a $R \in \mathrm{P}^A$ such that

$$x \in C \leftrightarrow \exists y_1 \forall y_2 R(\langle x, y_1, y_2 \rangle)$$

The aim is to find a R' in P^L instead of P^A.

Since $A \in \mathrm{P}^L/\mathrm{Poly}$, let a_n be polynomial-size advice for strings of length $\leqslant n$. Let A' be the set in P^L such that, for $|x| \leqslant n$,

$$x \in A \leftrightarrow \langle x, a_n \rangle \in A'.$$

Let $M_{A'L}$ be an oracle Turing machine accepting A' using oracle L. This machine exists because $A \in \mathrm{P}^L$.

Let M_{RA} be an oracle Turing machine accepting R using oracle A. This machine exists because $R \in \mathrm{P}^A$.

Define a machine M_{RaL} that will attempt to accept R, using part of its input as advice and its oracle L as follows. On input $\langle x, y_1, y_2, a \rangle$, M_{RaL} simulates M_{RA} on $\langle x, y_1, y_2 \rangle$. Whenever M_{RA} makes a query z to its oracle, M_{RaL} simulates $M_{A'L}$ on $\langle z, a \rangle$ to answer it. M_{RaL} uses its own oracle to answer queries of $M_{A'L}$.

Note that if $a = a_n$ for large enough n, say $n > p(|x|)$ for a polynomial p majorizing the time taken by M_{RaL}, then

$$M_{RaL}^L \text{ accepts } \langle x, y_1, y_2, a_n \rangle \leftrightarrow \langle x, y_1, y_2 \rangle \in R.$$

Now we have

$$x \in C \leftrightarrow (\exists \langle n, w \rangle : n > p(|x|))\ [w = a_n \text{ and } \exists y_1 \forall y_2 (M_{RaL}^L \text{ accepts } \langle x, y_1, y_2, w \rangle)] \tag{2}$$

The second part of the "and" is a $\Sigma_2^{\mathrm{P},L}$ predicate. We cannot show that $w = a_n$ is also a $\Sigma_2^{\mathrm{P},L}$ predicate, but we will show that it can be replaced by such a predicate.

For checking that an advice w is suitable, we only need that it is good enough to make $M^{L}_{A'L}$ simulate oracle A correctly on all the queries of M^{L}_{RaL} on $\langle x, y_1, y_2, w\rangle$. We check this by checking that the answers of $M^{L}_{A'L}$ conform to the self-reducibility of A. To be more precise, let M_{sr} be an oracle Turing machine that implements the self-reducibility, i.e. M^{A}_{sr} accepts A by making only smaller queries to its oracle. Let $\leqslant_{\mathrm{o}}$ denote the OK partial order imposed by the self-reducibility.

Let $A(w)=\{z|\langle z, w\rangle$ is accepted by $M^{L}_{A'L}\}$. We show below that the test $w=a_n$ in (1) can be replaced by the following test of validity:

$$\langle n, w\rangle \text{ is } \textit{valid} \leftrightarrow (\forall z{:}|z|\leqslant n)(\forall z'\leqslant_{\mathrm{o}} z)$$
$$[z'\in A(w) \leftrightarrow z' \text{ is accepted by } M^{A(w)}_{\mathrm{sr}} \text{ and all queries generated by } M^{A(w)}_{\mathrm{sr}} \text{ on input } z' \text{ are } \leqslant_{\mathrm{o}} z'].$$

This predicate is in $\Pi^{\mathrm{P},L}_1 \subseteq \Sigma^{\mathrm{P},L}_2$.

We show first that, for every n, valid advice exists. Then we show that $M^{L}_{A'L}$ using valid advice simulates A correctly for the appropriate queries. This will complete the proof.

Consider the pairs $\langle n, a_m\rangle$. Let z be a string of size $\leqslant n$. The strings $z'\leqslant_{\mathrm{o}} z$ will have their size bounded by a polynomial. If m is large enough (but still polynomial in n), then $z'\in A \leftrightarrow z'\in A(a_m)$ for any of these strings z'. This means that $\langle n, a_m\rangle$ is valid.

Now, let x be a string, let $n\geqslant p(|x|)$ and let $\langle n, w\rangle$ be valid advice. The machine M^{L}_{RaL} on $\langle x, y_1, y_2, w\rangle$ simulates $M^{L}_{A'L}$ on strings of size $\leqslant n$. $M^{L}_{A'L}$ accepts $\langle z, w\rangle$ if and only if $z\in A(w)$, by definition of $A(w)$. We need $z\in A(w)\leftrightarrow z\in A$.

We show by induction on $\leqslant_{\mathrm{o}}$ that, for any string $z'\leqslant_{\mathrm{o}} z$,

$$z'\in A(w) \leftrightarrow z'\in A.$$

Suppose $\langle n, w\rangle$ is valid. If z' is a minimum string, then the self-reduction will make no query to the oracle. Then

$$\begin{aligned} z'\in A(w) &\leftrightarrow z' \text{ is accepted by } M^{A(w)}_{\mathrm{sr}} && \text{(by the validity of } w)\\ &\leftrightarrow z' \text{ is accepted by } M^{A}_{\mathrm{sr}} && \text{(because no query to oracle)}\\ &\leftrightarrow z'\in A && \text{(by self-reducibility).}\end{aligned}$$

Now, assume that $z''\in A \leftrightarrow z''\in A(w)$ for strings $z''\leqslant_{\mathrm{o}} z'$. The self-reduction will make queries only for strings smaller than z' to the oracle. Then again

$$\begin{aligned} z'\in A(w) &\leftrightarrow z' \text{ is accepted by } M^{A(w)}_{\mathrm{sr}} && \text{(by the validity of } w)\\ &\leftrightarrow z' \text{ is accepted by } M^{A}_{\mathrm{sr}} && \text{(because only smaller queries)}\\ &\leftrightarrow z'\in A && \text{(by self-reducibility).}\end{aligned}$$ □

It may be worth reexamining the subtleties in the definition of self-reducibility in the light of the proof of Theorem 3.10. We assume that all queries generated by the oracle Turing machine M_{sr} are smaller, in the OK ordering $\leqslant_{\mathrm{o}}$, than the input string

when M_{sr} is executed with A as the oracle. This assumption leaves open the possibility that execution of M_{sr} with some other oracle does not preserve the ordering. If the definition of Turing self-reducibility were strengthened to mean that every query generated is less than the input string, independent of the choice of oracle, then the proof of Theorem 3.10 would not require the OK ordering to be polynomial-time-recognizable. In particular, this kind of uniformity holds for $\leqslant_{tt}^{P}$-self-reducibility. Thus, if the assertion of Theorem 3.10 was restricted to $\leqslant_{tt}^{P}$-self-reducibility, polynomial-time recognizability of the OK ordering would not have been required.

This theorem has interesting corollaries. First we consider the consequences that occur when $L \in P$.

Corollary 3.11. *If A is self-reducible and A is $\leqslant_{T}^{P}$-reducible to a sparse set, then A is generalized low$_2$. In particular, $A \in \Sigma_2^P$. If $A \in NP$, A is self-reducible, and A is $\leqslant_{T}^{P}$-reducible to a sparse set, then A is low$_2$* [10].

Corollary 3.12. *If A is self-reducible and A is $\leqslant_{T}^{P}$-reducible to a p-selective set, then A is generalized low$_2$. If $A \in NP$, A is self-reducible, and A is $\leqslant_{T}^{P}$-reducible to a p-selective set, then A is low$_2$.*

This result is a nice contrast to the following observation: It is proved in [18] that a set B belongs to P if and only if it is both d-self-reducible and p-selective. Also, it is proved in [19] that if B is p-selective and $A \leqslant_{ptt}^{P} B$[1], then A is p-selective. Thus, if A is d-self-reducible and A is $\leqslant_{ptt}^{P}$-reducible to a p-selective set, then $A \in P$.

Next we consider the consequences of letting A be an NP-hard set.

Corollary 3.13. *If A is* NP*-hard (i.e. A is $\leqslant_{T}^{P}$-hard for NP) and $A \in P^L/Poly$, then L is generalized high$_2$.*

Proof. Let B be any NP-complete self-reducible set. Since $B \leqslant_{T}^{P} A$, B is in $P^L/Poly$. The theorem then says that $B R_2^P L$. Thus, L is *generalized high$_2$*. □

Corollaries 3.14 and 3.16 are the main results about promise problems in this paper.

Corollary 3.14. *If A is* NP*-hard, then every solution of* PP*-A is generalized high$_2$.*

Proof. Let B be a self-reducible complete set for NP. If L is a solution of PP-A, then $A \in P^L/Poly$. Since A is NP-hard, $B \in P^L/Poly$ as well. The corollary then follows. □

Corollary 3.15. *If A is* NP*-hard, $A \in P^L/Poly$,* and *L is generalized low$_2$, then the polynomial hierarchy collapses to Σ_2^P.*

[1] $\leqslant_{ptt}^{P}$ denotes polynomial-time positive truth-table reducibility. The reader may refer to [11, 18] for a definition.

Corollary 3.15 follows immediately from Corollary 3.13, for a set L cannot be both *generalized high*$_2$ and *generalized low*$_2$ unless the hierarchy collapses to Σ_2^{P}. Since a set cannot be both *generalized high*$_2$ and *generalized low*$_i$ unless the hierarchy collapses to Σ_i^{P}, we have the following corollary.

Corollary 3.16. *If A is* NP*-hard, then no solution of* PP*-A is generalized low$_i$, for any $i \geqslant 2$, unless the polynomial hierarchy collapses to Σ_i^{P}.*

The well-known result of Karp and Lipton [8] follows immediately by considering the case that $L \in P$.

Corollary 3.17 (Karp and Lipton [8]). *If* $\mathrm{NP} \subseteq \mathrm{P/Poly}$, *then the polynomial hierarchy collapses to Σ_2^{P}.*

Corollary 3.15 is even stronger than the following result that was first proved in [1].

Corollary 3.18 (Abadi et al. [1]). *If* $\mathrm{NP} \subseteq (\mathrm{NP} \cap \text{co-NP})/\mathrm{Poly}$, *then the polynomial hierarchy collapses to Σ_2^{P}.*

Proof. Noting that $\mathrm{NP} \cap \text{co-NP} = low_1$ and that $\mathrm{P}^{\mathrm{NP} \cap \text{co-NP}} = \mathrm{NP} \cap \text{co-NP}$, the corollary follows readily from Corollary 3.15. □

Next we state some corollaries that extend these results to arbitrary levels of the polynomial hierarchy.

Corollary 3.19. *Let $i \geqslant 1$. If A is a $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-hard set for Σ_i^{P} and $A \in \mathrm{P}^L/\mathrm{Poly}$, then L is generalized high$_{i+1}$.*

Proof. This follows from Theorem 3.10. For each i, it is well known that there are self-reducible complete sets for Σ_i^{P}. Let B be such a set. Then $B \leqslant_{\mathrm{T}}^{\mathrm{P}} A$ and, so, $B R_2^{\mathrm{P}} L$. Thus, $\Sigma_{i+2}^{\mathrm{P}} = \Sigma_2^{\mathrm{P},B} \subseteq \Sigma_2^{\mathrm{P},L} \subseteq \Sigma_{i+1}^{\mathrm{P},L}$. So, L is *generalized high*$_{i+1}$. □

Corollary 3.20. *Let $i \geqslant 1$. If A is a $\leqslant_{\mathrm{T}}^{\mathrm{P}}$-hard set for Σ_i^{P}, $A \in \mathrm{P}^L/\mathrm{Poly}$, and L is generalized low$_{i+1}$, then the polynomial hierarchy collapses to $\Sigma_{i+1}^{\mathrm{P}}$.*

Corollary 3.21. *For any L and for $i \geqslant 1$, if L is generalized low$_{i+1}$ and $\Sigma_i^{\mathrm{P}} \subseteq \mathrm{P}^{\Sigma_i^{\mathrm{P},L} \cap \Pi_i^{\mathrm{P},L}}/\mathrm{Poly}$, then $\Sigma_{i+2}^{\mathrm{P}} = \Sigma_{i+1}^{\mathrm{P}}$.*

Proof. Assume that the hypothesis is true and let A be a self-reducible complete set for Σ_i^{P}. Then $A \in \mathrm{P}^{\Sigma_i^{\mathrm{P},L} \cap \Pi_i^{\mathrm{P},L}}/\mathrm{Poly}$. So, there exists a set $L_1 \in \Sigma_i^{\mathrm{P},L} \cap \Pi_i^{\mathrm{P},L}$ such that $A \in \mathrm{P}^{L_1}/\mathrm{Poly}$. By Theorem 3.10, $A R_2^{\mathrm{P}} L_1$. Thus, $\Sigma_{i+2}^{\mathrm{P}} = \Sigma_2^{\mathrm{P},A} \subseteq \Sigma_2^{\mathrm{P},L_1} \subseteq \Sigma_{i+1}^{\mathrm{P},L}$. Thus, L is *generalized high*$_{i+1}$; so, the result follows from the hypothesis that L is *generalized low*$_{i+1}$. □

In [2] it is shown that the polynomial hierarchy does not collapse if and only if, for every sparse set S, the polynomial hierarchy relative to S does not collapse. Now we show that the principal tool used to obtain this result in [2] is also obtainable as a corollary to Theorem 3.10.

Corollary 3.22. (Balcázar et al. [2]). *If A is a self-reducible set and there is a $k \geqslant 0$ and a sparse set S such $A \in \Sigma_k^{\mathrm{P},S}$, then $\Sigma_2^{\mathrm{P},A} \subseteq \Sigma_{k+2}^{\mathrm{P}}$.*

Proof. Assume that the hypothesis holds and let L be any complete set for Σ_k^{P}. Then there is a sparse set S such that $A \leqslant_{\mathrm{T}}^{\mathrm{P}} L \oplus S$ and, so, $A \in \mathrm{P}^L/\mathrm{Poly}$. From Theorem 3.10 (letting $A = B$) $A R_2^{\mathrm{P}} L$ follows. This means that $\Sigma_2^{\mathrm{P},A} \subseteq \Sigma_2^{\mathrm{P},L} = \Sigma_{k+2}^{\mathrm{P}}$.

Acknowledgment

The authors are grateful to Ron Book for making them aware of similarities between this work and the elegant results in [2].

References

[1] M. Abadi, J. Feigenbaum and J. Kilian, On hiding information from an oracle, *J. Comput. System Sci.* **39** (1989) 21–50.
[2] J. Balcázar, R. Book and U. Schöning, The polynomial-time hierarchy and sparse oracles, *J. Assoc. Comput. Mach.* **33** (1986) 603–617.
[3] J. Balcázar, R. Book and U. Schöning, Sparse sets, lowness, and highness, *SIAM J. Comput.* **15** (1986) 739–747.
[4] S. Even, A. Selman and Y. Yacobi, The complexity of promise problems with applications to public-key cryptography, *Inform. And Control* **61** (1984) 159–173.
[5] S. Even and Y. Yacobi, Cryptocomplexity and NP-completeness, in: *Proc. 8th Colloq. on Automata, Langauges, and Programming*, Lecture Notes in Computer Science, Vol. 85 (Springer, Berlin, 1980) 195–207.
[6] M. Grötschel, L. Lovász and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* **1** (1981) 168–198.
[7] J. Grollmann and A. Selman, Complexity measures for public-key cryptosystems, *SIAM J. Comput.* **17** (1988) 309–335.
[8] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, in: *Proc. 12th ACM Symp. on Theory of Computing* (1980) 302–309.
[9] K. Ko, On self-reducibility and weak P-selectivity, *J. Comput. System Sci.* **26** (1983) 209–211.
[10] K. Ko and U. Schöning, On circuit-size and the low hierarchy in NP, *SIAM J. Comput.* **14** (1985) 41–51.
[11] R. Ladner, N. Lynch and A. Selman, A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1** (1975) 103–123.
[12] T. Long, Strong nondeterministic polynomial-time reducibilities, *Theoret. Comput. Sci.* **21** (1982) 1–25.
[13] A. Meyer and M. Paterson, With what frequency are apparently intractable problems difficult?, Technical Report MIT/LCS/TM-126, M.I.T., 1979.
[14] U. Schöning, A low and a high hierarchy within NP, *J. Comput. System Sci.* **27** (1983) 14–28.
[15] U. Schöning, Generalized polynomial reductions, degrees, and NP-completeness, *Fund. Inform.* **7** (1984) 77–843.

[16] A. Selman, Arithmetical reducibilities I, *Z. Math. Logik Grundlag. Math.* **17** (1971) 335–350.
[17] A. Selman, P-selective sets, tally languages, and the behavior of polynomial-time reducibilities on NP, *Math. Systems Theory* **13** (1979) 55–65.
[18] A. Selman, Reductions on NP and P-selective sets, *Theoret. Comput. Sci.* **19** (1982) 287–304.
[19] A. Selman, Promise problems complete for complexity classes, *Inform. and Comput.* **78** (1988) 87–98.
[20] L. Valiant and V. Vazirani, NP is as easy as detecting unique solutions, *Theoret. Comput. Sci.* **47** (1986) 85–93.